Your intuition is pointing at something fundamental: **gating doesn't just select computations—it becomes *self-reinforcing* through the geometry of weighted sums.** That has important consequences for specialization, stability, and generalization.

Let's unpack it cleanly.

---

1. The geometric core: weighted sums are directional filters

A linear unit computes a dot product:

- Large output ⇔ input aligns with the weight vector
- Small/negative output ⇔ input is misaligned

So each neuron is effectively:

> A **directional detector** that responds by magnitude to alignment.

When followed by ReLU (or any gating):

- Strong positive alignment → neuron turns "on"
- Weak or negative alignment → neuron turns "off"

This already creates a **winner-take-most effect** among directions.

---

2. Gating creates feedback on direction selection

Now introduce gating:

- Activation patterns determine which weights are used downstream
- Those downstream weights shape future representations

This creates a loop:

1. Input slightly aligns with some weight directions
2. Those neurons activate (gating decision)
3. Their outputs dominate the next layer
4. The next layer's weights adapt to those activated patterns
5. Future inputs are *more likely* to align with those same directions

So:

> **Small initial biases in direction get amplified into stable pathways.**

---

3. Reinforcement = formation of “preferred subspaces”

Over training, this leads to:

- Certain directions in representation space becoming **high-gain pathways**
- Others becoming suppressed or unused

In your hash-table view:

- Some “keys” (gating patterns) become **much more likely**
- Others effectively disappear

Geometrically:

> The network carves input space into **funnels** that guide inputs into a small number of stable activation patterns.

---

4. Why this matters: stability and consistency

This reinforcement is important because it creates:

(a) Stable routing

- Similar inputs consistently trigger the same gating pattern
- This makes the “hash lookup” reliable

Without this:

- Small perturbations would constantly flip gating bits
- The model would behave chaotically

(b) Coherent learning signals

Because the same gating patterns repeat:

- The same subsets of weights get updated together
- Learning accumulates rather than averaging out

So:

> Reinforcement turns noisy, high-dimensional data into **repeatable training contexts**.

---

5. It enables specialization of linear maps

Each gating pattern corresponds to a linear transformation.

Reinforcement ensures:

- A given pattern is activated by a **coherent set of inputs**
- Those inputs share structure (because they align with similar directions)

Therefore:

> Each linear map specializes to a *meaningful subset of the data*.

Without reinforcement:

- Each map would see random, inconsistent inputs
- It couldn't specialize effectively

---

6. Magnitude amplification sharpens decisions

You mentioned vector magnitude buildup—this is key.

As representations pass through layers:

- Aligned components get amplified
- Misaligned components get attenuated

This leads to:

- Increasingly **confident gating decisions** (far from zero)
- Reduced ambiguity near decision boundaries

So deeper layers tend to have:

- More **binary-like**, stable gating patterns

---

7. Implicit competition between pathways

Because of normalization constraints (explicit or implicit):

- Not all directions can grow equally

So reinforcement creates competition:

- Strong pathways get stronger
- Weak pathways fade

This is similar to:

- Feature selection
- Sparse coding
- Winner-take-all dynamics

---

8. Connection to generalization

This reinforcement mechanism helps generalization in a subtle way:

It groups inputs by *directional similarity*, not exact identity

- Inputs that differ in irrelevant ways but share direction:
  → trigger the same gating
  → use the same linear map

So the model learns:

> "If it points this way, treat it like this."

That's a strong inductive bias.

---

9. But there's a trade-off: path dependence

The same mechanism can also cause:

(a) Early commitment

- Random initial alignments get locked in

(b) Dead regions

- Some gating patterns never activate → never trained

(c) Reduced exploration

- The model may miss alternative useful partitions

So reinforcement is:

> Both a **stabilizer** and a **bias amplifier**

---

10. Putting it all together

Your idea can be summarized like this:

> Because weighted sums respond by magnitude to alignment,
> gating decisions preferentially select already-aligned directions.
> Training then reinforces those directions, creating stable activation patterns,
> which in turn define consistent “keys” for selecting linear mappings.

And the consequence:

> The network evolves into a system of **self-reinforcing routing pathways**,
> where each pathway corresponds to a directionally defined subset of input space
> and a specialized linear transformation.

---

11. A useful mental picture

Think of the network as:

- A landscape of valleys (preferred directions)
- Inputs “fall” into the nearest valley
- Each valley routes to a specific linear operator
- Training deepens frequently used valleys

If you want, I can connect this directly to:

- mixture-of-experts routing stability,
- attention mechanisms (which do something similar but softly),
- or formalize it into a short theorem-style statement like before.